# Sandwich Approximation of Univariate Convex Functions with an Application to Separable Convex Programming

**Rainer E. Burkard**
*Institut für Mathematik, Technische Universität Graz, Kopernikusgasse 24, A-8010 Graz, Austria*

**Horst W. Hamacher**
*Universität Kaiserslautern, Fachbereich Mathematik, Erwin-Schrödinger-Strasse, D-6750 Kaiserslautern, Germany*

**Günter Rote**
*Institute für Mathematik, Technische Universität Graz, Kopernikusgasse 24, A-8010 Graz, Austria*

In this article an algorithm for computing upper and lower $\varepsilon$ approximations of a (implicitly or explicitly) given convex function $h$ defined on an interval of length $T$ is developed. The approximations can be obtained under weak assumptions on $h$ (in particular, no differentiability), and the error decreases quadratically with the number of iterations. To reach an absolute accuracy of $\varepsilon$ the number of iterations is bounded by $\sqrt{9DT/8\varepsilon}$, where $D$ is the total increase in slope of $h$. As an application we discuss separable convex programs.

## 1. INTRODUCTION

Convex functions play an important role in mathematical programming. Many models lead directly to convex functions, or they arise as value functions of parametric linear programs, in time/cost trade-off problems, or in multicriteria optimization. There are several reasons for replacing a convex function by a piecewise linear approximation with few breakpoints: In some models piecewise linear functions are easier to handle, for example, in separable convex programming. This will be utilized in Section 3. Another reason is that the evaluation of a convex function $h(t)$ for a given parameter $t$ may be costly. Therefore one is interested in getting an approximation of this function with as few function evaluations as possible. This arises, for instance, if $h(t)$ is the value function of a parametric linear program: Evaluating $h(t)$ amounts to solving a linear program. The same situation occurs in the context of bicriteria linear programs, since the efficient point curve is convex. The special case of bicriterial minimum cost network-flow problems has been investigated in Fruhwirth, Burkard, and Rote [6].

Whereas in convex programming a locally good approximation in the neighborhood of the optimal solution is sufficient, a global approximation with a uniform error bound on the whole range is required in the other applications mentioned above: In bicriteria and parametric programming one is interested in the optimal value function (or efficient point curve, respectively) as a whole.

Several authors have considered the problem of approximating convex functions $h(t)$ by a piecewise linear function. If $h$ is twice continuously differentiable ($h \in C^2$), Phillips [13] finds approximations with a given error bound on the whole domain of $h$. Phillip's procedure requires the computation of roots of nonlinear equations to find the points at which the piecewise approximation and the original function coincide. (These points will subsequently be called knots). By using this procedure iteratively one can also find the best approximation with respect to a given number of knots. Cox [4] showed that Phillip's results hold if $h$ is just continuously differentiable ($h \in C^1$), but that the computations of the knots can be simplified if $h \in C^2$. Thakur's [20–22] objective is to find approximations such that the difference between the optimal objective values of the original mathematical program and its approximate version is bounded by a given error. He solves a series of smaller piecewise linear problems and uses an equidistant partitioning of the domain of $h$ to develop convergent procedures. Salem and Elmaghraby [17] develop local approximation methods for convex, decreasing, but nonquadratic functions, and apply their results to project networks. Further references for approximation by piecewise linear functions include Kao and Meyer [10], Sonnevend [18, 19], and the textbooks of Bazaraa and Shetty [2] and Rockafellar [14]. Geoffrion [7] develops a more general theory of approximating objective functions in mathematical programming.

The geometric problem of approximating a plane convex figure by a convex polygon, which is essentially the same as our problem, has also received great attention in the literature, both from a geometric theoretical view point (cf. the overview of Gruber [8]) as well as in the applied areas of image processing and computational geometry. A nice overview of algorithms from this area is given in Kurozumi and Davis [11], cf. also Rote [16].

In this article we develop the sandwich algorithm for computing a global approximation for a convex function $h$ in a given interval $[a, b]$. At any stage of the algorithm piecewise-linear lower and upper approximations $l(t)$ and $u(t)$, respectively, of $h(t)$ are known on the whole domain $[a, b]$ of $h$ and will satisfy

$$l(t) \leq h(t) \leq u(t), \qquad \text{for all } a \leq t \leq b.$$

The algorithm terminates when the global error is less than a prespecified $\varepsilon$; i.e.,

$$\sup_{a \leq t \leq b} \{u(t) - l(t)\} < \varepsilon.$$

Alternatively, the sandwich algorithm may be terminated if a given number of knots have been computed. In this case one will be able to compute a priori an achievable error bound of the approximation.

Sandwiching is a very natural process that has been proposed in different contexts and with different variations by various authors, like Aneja and Nair [1979] or some of the articles cited above. The contribution of our article is mainly theoretical, by giving an error analysis: After presenting the sandwich algorithm (in Section 2), we will show that, for a given bound $\varepsilon$, the number of necessary iterations is bounded by $\frac{3}{2}\sqrt{D(b-a)/2\varepsilon}$, where $D$ is the total increase in slope of $h$ on the interval $[a, b]$. Each iteration amounts essentially to an evaluation of $h(t)$ and two (one-sided) derivatives. A similar approach was developed by Sonnevend [19] in a more general setting. Sonnevend uses an inductive argument for establishing that the number of iterations is $O[\sqrt{D(b-a)/\varepsilon}]$. Our proof is constructive, uses combinatorial arguments, and yields the best possible constant in the iteration bound.

In Section 3, we describe as a simple application the approximative solution of separable convex programs. Concluding remarks and some lines for further research are given in the last section.

## 2. THE SANDWICH ALGORITHM: APPROXIMATION OF CONVEX FUNCTIONS

### 2.1. Preliminaries

Let $h: [a, b] \to \mathrm{R}$ be a convex function, defined on a bounded interval $[a, b] \subset \mathrm{R}$. We assume that $h$ is continuous at the endpoints of the interval and that for any $t \in [a, b]$ the left and right derivative of $h$ is available (or can be computed). Moreover, the one-sided derivatives should be finite in the endpoints of $[a, b]$.

We want to compute efficiently two piecewise-linear, convex functions $l(t)$ and $u(t)$ such that

$$l(t) \leqq h(t) \leqq u(t) \quad \text{and} \quad u(t) - l(t) \leqq \varepsilon, \qquad \text{for all } t \in [a, b].$$

The idea for constructing $l(t)$ and $u(t)$ is as follows. Let $a = t_0 < t_1 < t_2 < \cdots < t_n = b$ be a finite partition of the interval $[a, b]$. For any $t_i$ ($i = 0, 1, \ldots, n-1$) let $h_i^+$ be the right derivative of $h$ at $t_i$ and let $h_i^-$ be the left derivative of $h$ at $t_i$ ($i = 1, 2, \ldots, n$). Then $l(t)$ and $u(t)$ are defined as follows:

$$u(t) := h(t_i) + \frac{h(t_{i+1}) - h(t_i)}{t_{i+1} - t_i} \cdot (t - t_i)$$

and

$$l(t) = \max\{h(t_i) + h_i^+ \cdot (t - t_i),\ h(t_{i+1}) + h_{i+1}^- \cdot (t - t_{i+1})\}$$

$$\text{for} \quad t_i \leqq t \leqq t_{i+1}, \qquad i = 0, 1, \ldots, n-1.$$

The definition of $l$ and $u$ is illustrated in Figure 1. It should be noted that $h_i^+$ as well as $h_i^-$ can be replaced by any subgradient $\delta_i$ of $h$ at $t_i$. Whereas the worst-case bound developed in the following is independent of the choice of $\delta_i$, $h_i^+$ and $h_i^-$ are certainly preferable, since they yield the tightest lower bounds.

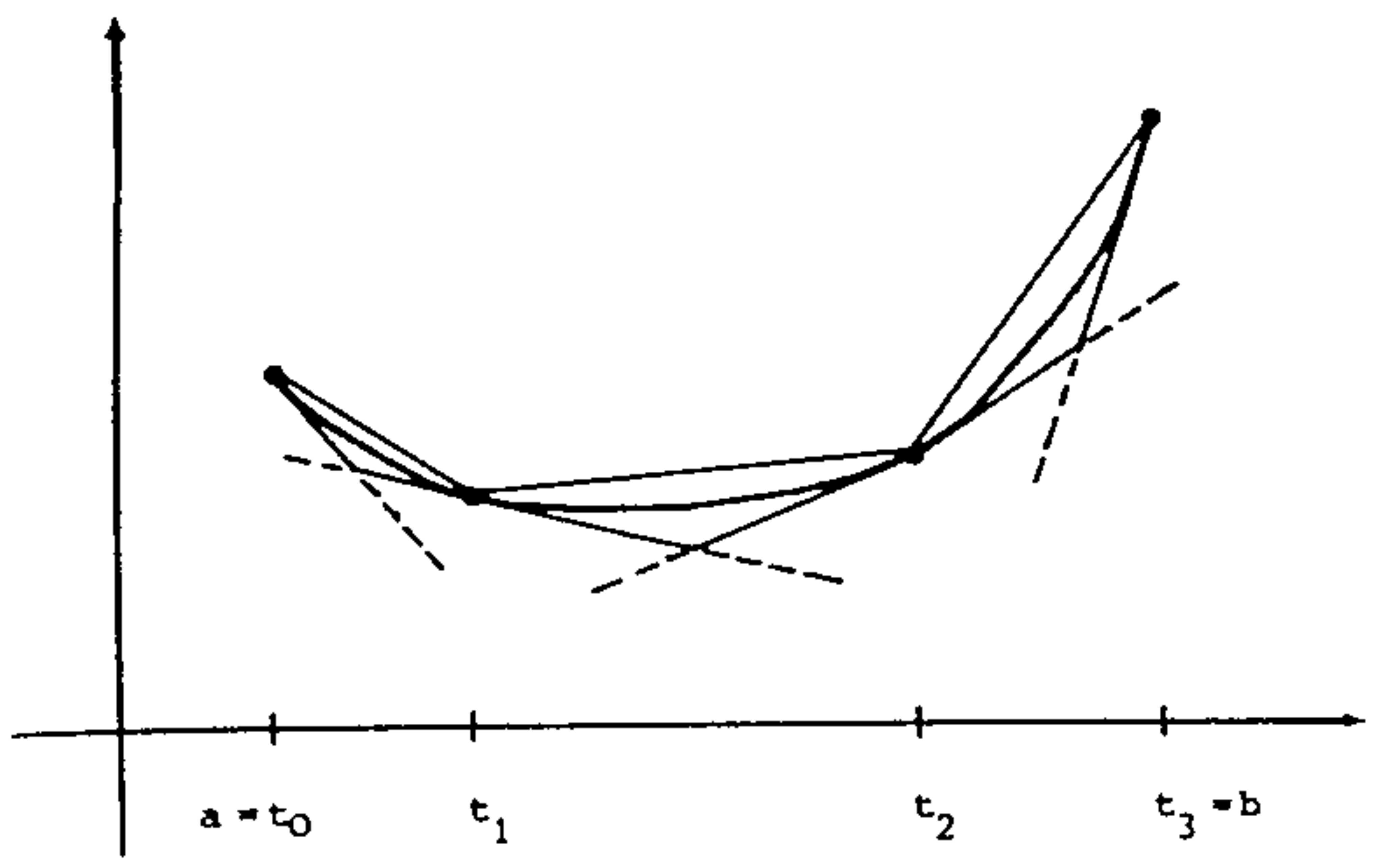

**Figure 1.** A convex function $h(t)$ with lower approximation $l(t)$ and upper approximation $u(t)$.

It follows from the definitions that

- $l(t)$ and $u(t)$ are piecewise linear and convex
- for all $t \in [a, b]$: $l(t) \leqq h(t) \leqq u(t)$.

In any interval $[t_i, t_{i+1}]$ the maximal difference between $u(t)$ and $l(t)$ is attained at the point $t_i^*$, defined by

$$h(t_i) + h_i^+(t_i^* - t_i) = h(t_{i+1}) + h_{i+1}^-(t_i^* - t_{i+1}),$$

i.e., $t_i^*$ is uniquely defined, if $h_i^+ < h_{i+1}^-$, namely, by

$$t_i^* = \frac{h(t_{i+1}) - h(t_i) + h_i^+ t_i - h_{i+1}^- t_{i+1}}{h_i^+ - h_{i+1}^-}.$$

If $h_i^+ = h_{i+1}^-$, then $l(t) = h(t) = u(t)$ for all $t_i \leqq t \leqq t_{i+1}$. Therefore the maximal error $E := \max_{a \leq t \leq b}(|h(t) - u(t)|, |h(t) - l(t)|)$ is bounded by

$$\max_{0 \leq i \leq n-1} \{u(t_i^*) - l(t_i^*)\} = \max_{0 \leq i \leq n-1} E_i,$$

where

$$E_i = \left[\frac{h(t_{i+1}) - h(t_i)}{t_{i+1} - t_i} - h_i^+\right] \cdot (t_i^* - t_i). \tag{1}$$

The following lemma (cf. Thakur [20, Lemma 2]) shows a relation between the length of an interval $I = [t_i, t_{i+1}]$, the one-sided derivatives in the endpoints of $I$ and the error $\max_{t \in I}(u(t) - l(t))$.

## 2.3. The Convergence Rate of the Sandwich Algorithm

We shall analyze in the following the two bisection rules and we show that they guarantee quadratic convergence of $l(t)$ and $u(t)$ to $h(t)$. For the maximum error rule, the proof methods of this article cannot be applied. However, by using different techniques, the quadratic convergence of the Sandwich algorithm with the maximum error rule has been established recently by Rote [16]; cf. also Rote [15].

The treatment of the two bisection rules is very similar. We shall first deal with the interval bisection rule.

The bisection algorithm can be visualized on a binary tree (Figure 3). The root of the tree corresponds to the interval $[a, b]$. Every inner node of the tree corresponds to an interval in which the error bound is not met and has exactly two successors. When the sandwich algorithm terminates, every leaf corresponds to an interval in which the error is not greater than $\varepsilon$.

In each iteration the algorithm picks a leaf of the tree with associated interval $[t_i, \bar{t}_i]$ in which the error exceeds $\varepsilon$. Then we set $\tilde{t} = \frac{1}{2}(\bar{t}_i - t_i)$, we compute $h(\tilde{t})$ and the one-sided derivatives $h^+(\tilde{t})$ and $h^-(\tilde{t})$, and we add two successors corresponding to $[t_i, \tilde{t}]$ and $[\tilde{t}, \bar{t}_i]$ to the tree.

Let $\hat{\tau}$ be the tree at the end of the algorithm. If $M$ is the number of evaluations of $h(t)$ and its one-sided derivatives $h^+(t)$, $h^-(t)$, then $\hat{\tau}$ has $M$-2 inner nodes and $M$-1 leaves. When we remove all leaves from $\hat{\tau}$ we get a binary tree $\tau$ with $M$-2 nodes. Let $v_i$ be the number of leaves of this new tree $\tau$ at level $i$. Now we shall make use of the following lemma on binary trees, which will be shown later.

LEMMA 2.2: If a binary tree $\tau$ has $n - 1$ nodes ($n \geqq 2$) and the number of its leaves at level $i$ ($i \geqq 0$) is $v_i$, then

$$w(\tau) := \sum_{i \geqq 0} 2^i v_i \geqq \frac{2}{9} \cdot n^2$$

and this bound is tight for infinitely many $n$.

If we denote $T := b - a$, any node at level $i$ corresponds to an interval $[t_i, \bar{t}_i]$

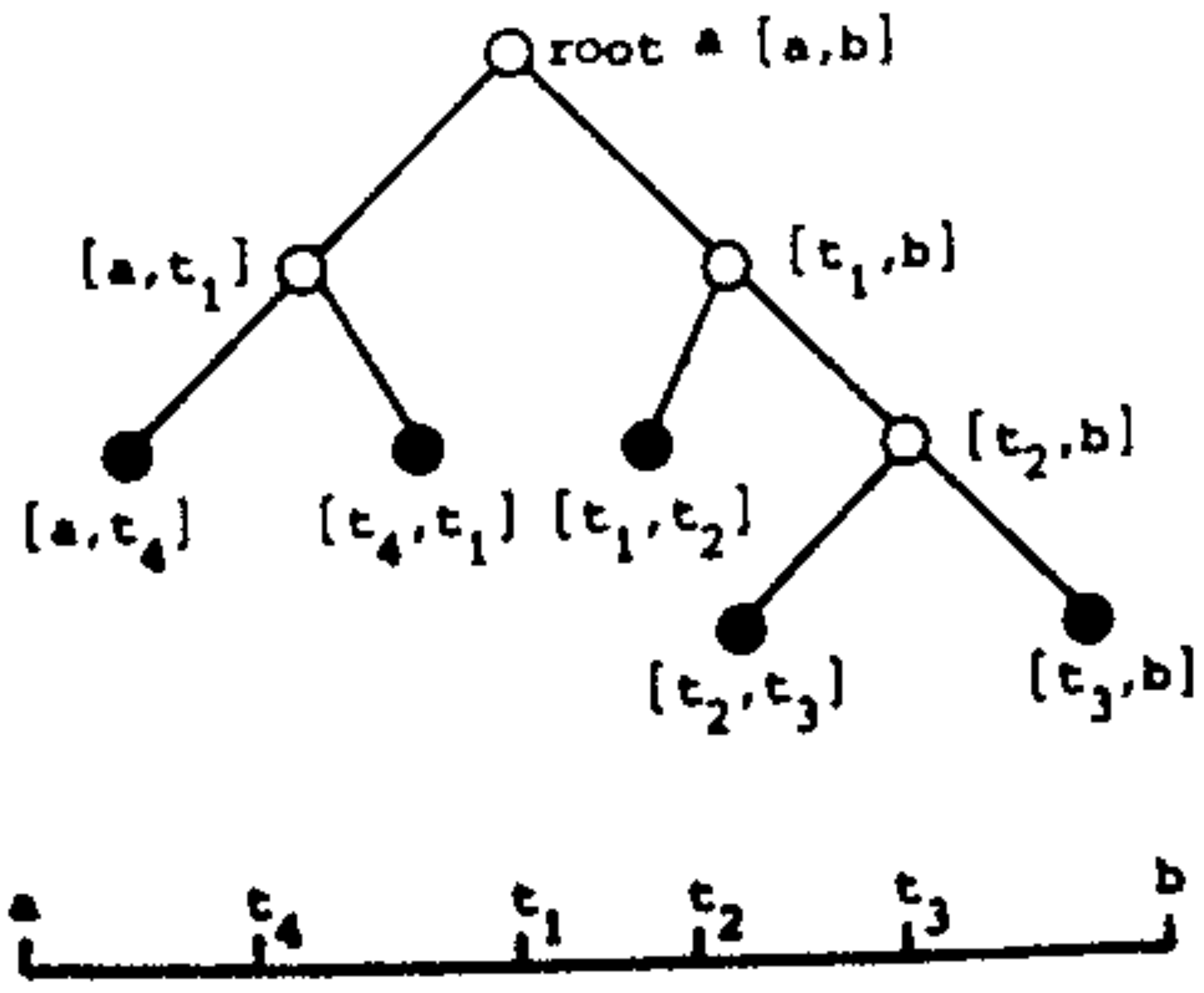

**Figure 3.** A bisection tree.

with length $T \cdot 2^{-i}$. Thus Lemma 2.1 yields

$$h^-(\bar{t}_j) - h^+(t_j) > \frac{4\varepsilon}{\bar{t}_j - t_j} = \frac{4\varepsilon 2^i}{T}.$$

Therefore we get for $M > 2$ (i.e., $\tau$ nonempty)

$$h^-(b) - h^+(a) \geqq \sum_{\text{leaves of } \tau} [h^-(\bar{t}_j) - h^+(t_j)],$$

since the intervals corresponding to the leaves are disjoint (except for common endpoints); the one-sided derivatives are nondecreasing and $h^-(t) \leqq h^+(t)$ for all $t$.

Hence, by Lemma 2.2, we get

$$h^-(b) - h^+(a) > \sum_{i \geqq 0} v_i \frac{4\varepsilon \cdot 2^i}{T} = \frac{4\varepsilon}{T} \cdot \sum_{i \geqq 0} 2^i v_i \geqq \frac{8}{9} \frac{\varepsilon}{T} (M - 1)^2.$$

Thus we have shown the following theorem for the interval bisection rule.

THEOREM 2.3: The number $M$ of evaluations of $h(t)$, $h^+(t)$, and $h^-(t)$ needed to obtain an upper and lower $\varepsilon$ approximation of $h(t)$ by the sandwich algorithm with interval bisection or with slope bisection is bounded as follows:

$$M \leq \max\left(2, \left\lceil \frac{3}{2} \sqrt{\frac{T}{2\varepsilon} \cdot (h^-(b) - h^+(a))} \right\rceil\right).$$

We can express this conversely by saying: If we use $M$ evaluations of $h(t)$, $h^+(t)$, and $h^-(t)$ and always choose the interval with the largest error to be partitioned next, then the lower and upper approximations $l(t)$ and $u(t)$ fulfill $l(t) \leq h(t) \leq u(t)$ and

$$\max_{a \leq t \leq b} (u(t) - l(t)) \leq \frac{9}{8} \frac{T}{(M - 1)^2} [h^-(b) - h^+(a)]$$

$$= \frac{K}{(M - 1)^2}, \qquad K \text{ constant.}$$

In case of the slope bisection rule, we simply have to exchange the role of $h^-(\bar{t}_j) - h^+(t_j)$ with $\bar{t}_j - t_j$. The proof is then completely analogous: We know that the slope difference at level $i$ is at most $(h^-(\bar{t}_j) - h^+(t_j))/2^i$, and Lemma 2.1 gives us then a lower bound for $\bar{t}_j - t_j$. We then sum $\bar{t}_j - t_j$ over all leaves, which gives $T = b - a$, and the theorem follows in the same way as above.

What remains to be shown is Lemma 2.2.

PROOF (of Lemma 2.2): Let $\mathcal{T}$ be a binary tree with height $d$ and let

$$w(\mathcal{T}) := \sum_{i=0}^{d} 2^i v_i,$$

where $v_i$ is the number of leaves of $\mathcal{T}$ at level $i$. At first we change $\mathcal{T}$ by applying the following two transformations, as long as they are possible ($d$ always denotes the height of the current tree $\mathcal{T}$):

*Transformation I* (cf. Figure 4): If there is a node $y$ at some level $i \leqq d - 2$, which has at most one successor, then we remove an arbitrary node $x$ from level $d$ and make it a successor of $y$.

By transformation $I$, $w$ decreases by $2^d$ by removing $x$ and increases by at most $2^{d-1}$ by inserting $x$. Moreover, $w$ may additionally increase by $2^{d-1}$, if the predecessor of $x$ becomes a leaf in $\tau'$. Thus we get in any case

$$w(\tau') \leqq w(\tau).$$

*Transformation II* (cf. Figure 5): If there is a node $z$ at level $d - 1$ with two successors $x$ and $y$ and another node $u$ at level $d - 1$ which is a leaf, then we make $y$ a successor of $u$.

By transformation II $w(\tau)$ decreases by $2^{d-1}$, because $u$ ceases to be a leaf.

When we perform these two transformations as long as they are possible, we arrive finally at a tree $\hat{\tau}$ with the following properties:

- $\tau$ and $\hat{\tau}$ have the same number of nodes;
- $w(\hat{\tau}) \leqq w(\tau)$;
- $v_i(\hat{\tau}) = 0$ for $i \leqq d - 2$;
- every node of $\hat{\tau}$ at level $i$, $i \leqq d - 2$, has exactly two successors;
- there are two cases:
  (a) all nodes at level $d - 1$ have at most one successor, or
  (b) all nodes at level $d - 1$ have one or two successors.

Since we consider from now on only the tree $\hat{\tau}$, we abbreviate $d = d(\hat{\tau})$ and $v_i = v_i(\hat{\tau})$. The nodes in $\hat{\tau}$ from level 0 to $d - 1$ form a complete binary tree with $2^d - 1$ nodes. At level $d - 1$ there are exactly $2^{d-1}$ nodes and there remain $(n - 1) - (2^d - 1) = n - 2^d$ nodes for level $d$.

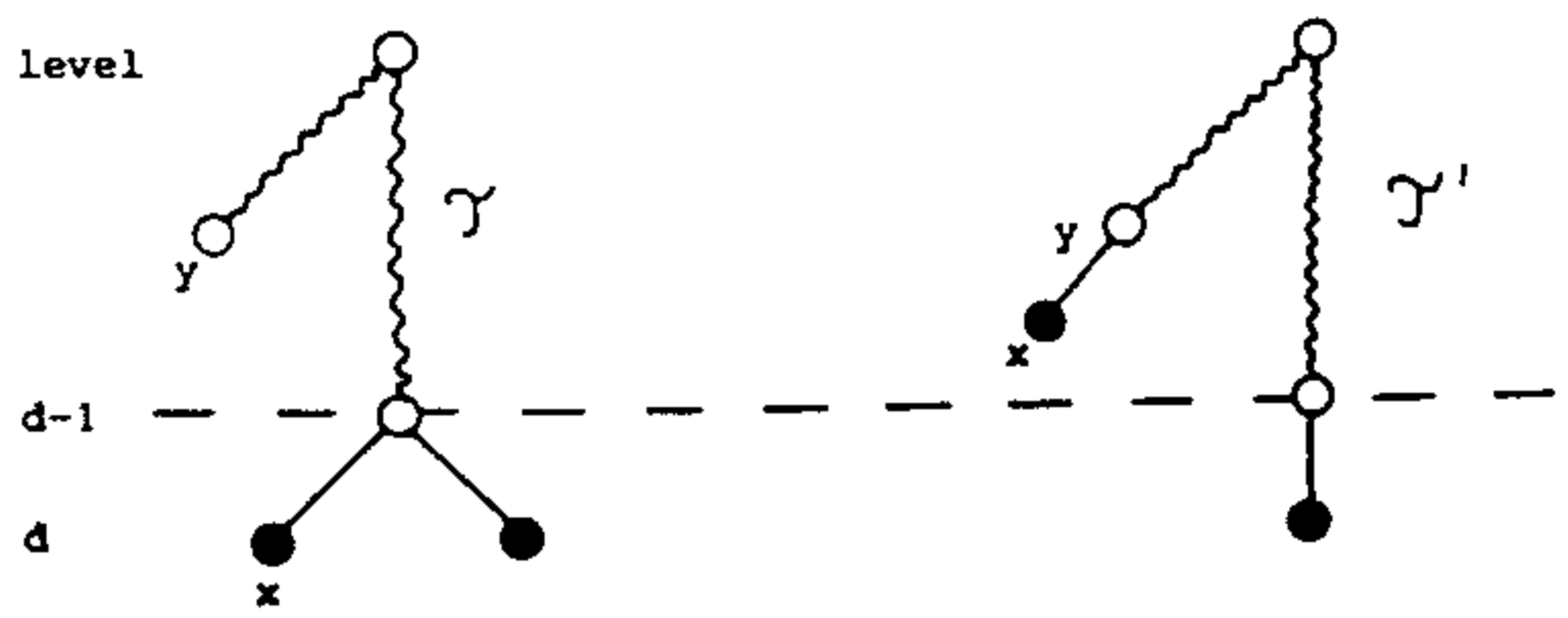

**Figure 4.** Illustration of transformation I.

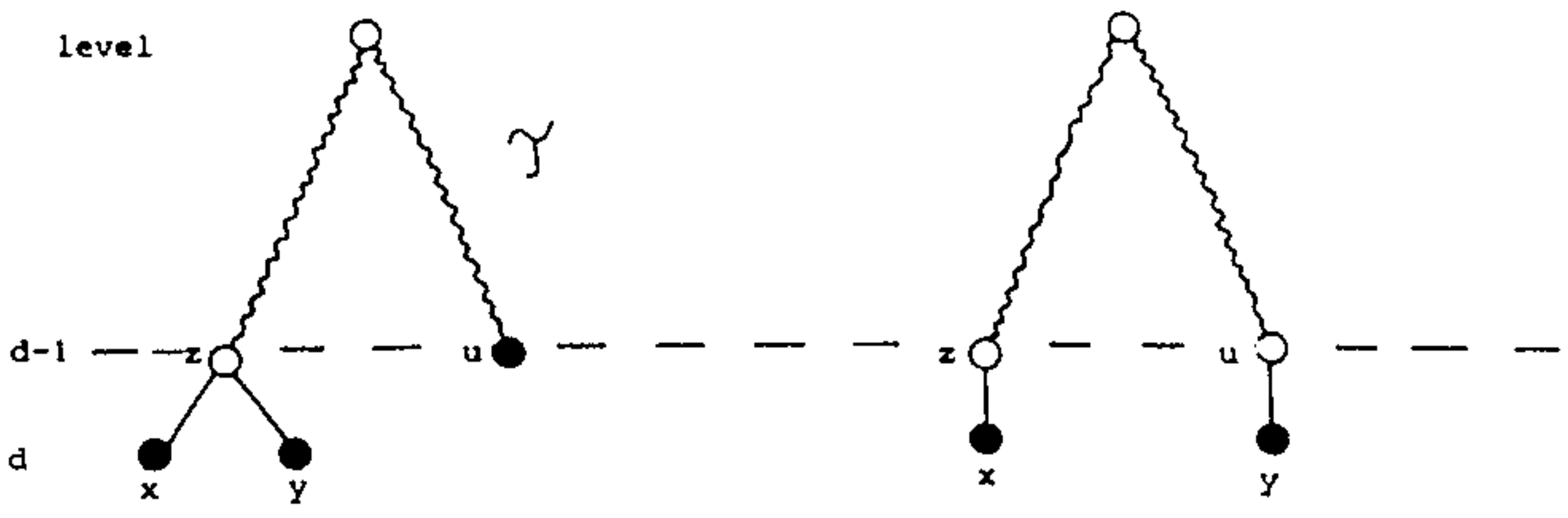

**Figure 5.** Illustration of transformation II.

Therefore we get

$$\text{Case A:} \quad v_{d-1} = 2^{d-1} - (n - 2^d),$$

$$w(\hat{\tau}) = n \cdot 2^{d-1} - 4^{d-1}.$$

$$\text{Case B:} \quad v_{d-1} = 0$$

$$w(\hat{\tau}) = n \cdot 2^d - 4^d.$$

But in Case A, $0 \leqq n - 2^d \leqq 2^{d-1}$ holds, which yields

$$2^d \leqq n \leqq 2^d + 2^{d-1},$$

whereas in Case B, $2^{d-1} \leqq n - 2^d \leqq 2^d$ implies

$$2^{d-1} + 2^d \leqq n \leqq 2^{d+1}.$$

Thus we get for $n$ with

$$2^d \leqq n \leqq 2^d + 2^{d-1} \quad \text{(Case A):} \quad w(\hat{\tau}) = 2^{d-1} \cdot n - 4^{d-1} =: w_A(n).$$

$$2^d + 2^{d-1} \leqq n \leqq 2^{d+1} \quad \text{(Case B):} \quad w(\hat{\tau}) = 2^d \cdot n - 4^d =: w_B(n).$$

In case $n = 2^d + 2^{d-1}$ (i.e., every node at level $d - 1$ has exactly one successor) both expressions $w_A(n)$ and $w_B(n)$ have the same value.

These two expressions $w_A(n)$ and $w_B(n)$ are tight lower bounds on $w(\hat{\tau})$ for a tree $\hat{\tau}$ with $n - 1$ nodes. To conclude the proof, we have to show that $w_A(n) \geqq \frac{2}{9}n^2$ and $w_B(n) \geqq \frac{2}{9}n^2$. Since both expressions are linear functions of $n$ it suffices to consider the boundary cases $n = 2^d$, $n = 2^d + 2^{d-1}$ and $n = 2^{d+1}$.

In the first case, $n = 2^d$, we get

$$w_A(n) = 2^{d-1}n - 4^{d-1} = \frac{n}{2}n - \left(\frac{n}{2}\right)^2 = \frac{n^2}{4} \geqq \frac{2}{9}n^2.$$

In the second case, $n = 2^d + 2^{d-1}$, we have $2^d = \frac{2}{3}n$. Therefore

$$w_A(n) = w_B(n) = 2^d n - 4^d = \tfrac{2}{3}n^2 - (\tfrac{2}{3}n)^2 = \tfrac{2}{9}n^2,$$